Algorithms for Big Data
Prof. John Ebenezer Augustine
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 47
MapReduce

So, today's and tomorrow's class, we are going to change gears a little bit by looking at topics that are sort of up and coming and, but still related to the foundations that you have seen of distribute algorithms. So, one obviously, everybody will be attest to this big data, obviously, one of the, like popular teams today everybody works probably in one aspect of big data or the other or at least people like to say that I work they connect whatever they do they connected with big data because it helps them.

So, today we are going to look at distributed big data that means, distributed algorithms of big data. So, it is not a sort of a fake connection, but a very real connection. So, distributed algorithms and big data they go together very well, I mean there are different aspects of big data, people are studying, for example, people can look at machine learning for big data, mining big data and so on and so forth. Here, we are looking at an orthogonal aspect which is how do you do distribution computation because in a sense I think that is probably one of the key aspects, if not the key aspect of computing with big data. So, the main philosophy in big data is that, at least one of the main philosophy is that we really drawing about really big data that means, data that cannot fit into one machine.

So big that you cannot load all the data in your main memory in one laptop and do some computation. If you can do that still many data that people say big data can be done like that I mean because our laptop become, some more laptops, some other higher end laptops they have like even terabyte of memory probably and so, you can do it at least external. So, you can practice you can still run them, but if the data keeps growing there is always going to be some limit. So, at least theoretically, that is we do not mean we are not interested in that scenario.

So theoretically, the scenario that big data computations model are those situations where the data is so large like, for example, a very good example is the web graph, the web

graph is. So, huge think maybe it is now we clearly the billions of nodes. So, what is the node in the web graph? Each node is the web page and that is the link from one web page to the other, if there is a hyperlink. So, this is the graph that Google users do all the things, I mean it is the main data structure that Google users to earn its own money. So, such everything it is happening on this web graph. So, billions of nodes and trillions of edges and it does a lot of processing on that, I mean computing page ranks and what not. So, huge that you cannot really try to do any. I mean you can fit it that is how your aspect I mean it is. So, I mean it the computation will be slowed down.

So, that is really the aspect, even the data can be fitted into the machine. It is still sequentially computing data, it is too costly. So, you want to partition their data across many machines and then jointly compute the link, so that is the whole philosophy of distributed computation. Big companies like Google, Facebook and even so many companies and mean even many, many companies that have that are not like common knowledge like Google, Facebook, many even smaller companies that you have never heard of, they have now clusters, the machines doing computation on that data that is what they people call now, data analytic something like that.

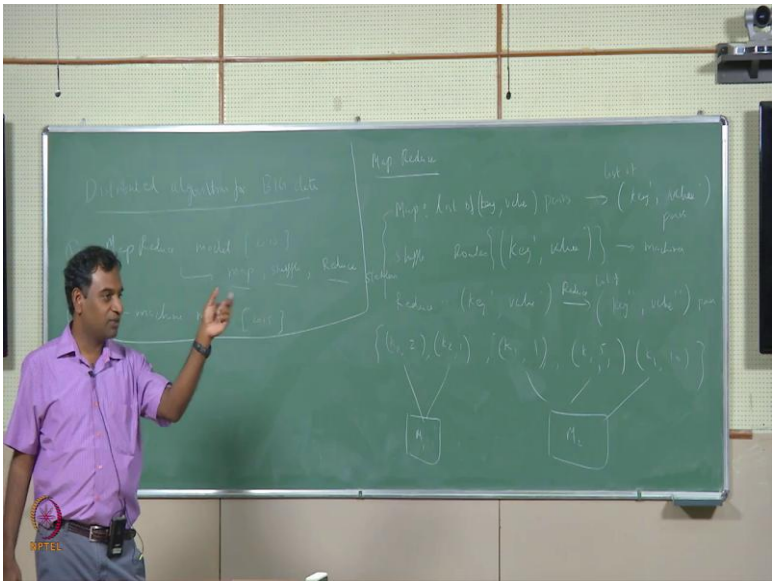
So, really we want understand how one can do computation on large scale data in a distributed way that is what we are going to study. So, we are going to study more theory for it, but it is not like many of the distributed algorithms that have that we have seen till now, it is not difficult to take these algorithms are trying to implement now. In fact, these algorithms I like more because they are much more amenable to implementation.

Then some of the algorithms that have seen before just for the simple fact that like we have seen MST algorithms and things like that, to really implement them. I mean, you can simulate them, but if you want to really implement them you have to really implement them on a communication network and you have to get hold of a network. There are such network simulators that you can do, but it is not that it is not like programming your sorting algorithm, study the performance of a sorting algorithm in your laptop. It is not as simple as like that then implementing distributed algorithms.

But these algorithms are somewhat more amenable. So, you can take many of the software packages that are available like MPI or Spark dupe, all those things and you can implement them and study them basically. Since you are interested in theory, we want to

study at the theory for this and try to again see how you can decide good algorithms rather than just willy-nilly design algorithms. The two things, I am going to talk about today, I would say that two important models that capture the distributed computing on big data. So, two are the probably important models today and one is the map reduce model.

(Refer Slide Time: 06:06)



We are going to see that and the second is I would say the k-machine model. The map reduce model is the model that models the map reduce computation that is popularly people say that dupe and all dupe job and whatever, but we are going to see the theory behind it. So, there is no a well established theory. Similarly, the k-machine model is a model, we will see why this model is more suitable for especially graph computation, but even for I would say it is even a better model to capture generally distributed computation on large data, especially large graph data than map reduce, but historically map reduce was the first model.

So, map reduce model was introduced some time in 2010. So, this model was introduced in 2015, I was also part of this model. So, I am little bit going to talk about one model, but it is now getting one more adapted by other people. So, we will see why this model is better and what it captures, but first let us understand map reduce model which is somewhat more well established because been heavier for a while and also although the popularity of map reduce is the sort of waning, I would say, in sense like I heard that

Google is using less of map reduce and more of what are, what I call this message passing kind of distributed computation.

I will come to that second and that is what does this model captures the k-machine, but let us first understand map reduce model and of course, map reduce was popular still popular, but it does really popular a few years back, like it was sort of the de facto standard for doing large scale computing. So, what is the map reduce model? As you know if I have done like this kind of dupe code and things like that, the map reduce is basically is an abstraction for I am going to abstract out the process of doing parallel computing. So, that is if the one of the probably the main contribution of map reduce.

So, it is parallel computing model, but it is such simplified parallel computing model. I do not know you have seen other parallel computing models like what? P RAM, but this is the much, much simplified model and that is why it is arguably easier to implement and that is why, it won many awards, the inventors of map reduce they won many awards because two people at Google. So, it is much simpler, but you can show that is generally it has essentially the same power as if you take enough rounds it will take, it has the same power as the P RAM model, but it is much simpler, so easy to program. So, why it is so? Simply, because it does basically only two operations, three operations, but essentially two operations; one is completely sort of transparent to the user map shuffle and reduce.

What are these operations? So, you can think over map reduce algorithm as going in rounds and each round has these three phases one after the other. The three consecutive phases map shuffle reduce and then map shuffle reduce and so on. So, what is the map operation? Map operation is nothing, but some operation it is a very simple, it has a very simple sort of api or input output relationship, it takes a bunch of, it takes an input which is basically always you can characterize them as key value pairs. So, it takes a list of key value pairs, basically tuples. So, map basically they, let us focus on.

So, I am going to talk about map reduce. So, basically takes a list of key value pairs that is the input. So, it is a fun, you think of it is a function. So, function that takes a list of or rather a mapping it takes a list of key value pairs and outputs another list of key value pairs some other list these can be in multi set meaning that they can be the same, it is not they are not. So, you can think of them as lists, there can be duplicates. It is not really a

set. A map takes a multi set of key value pairs and output is a multi set of key value pairs some other multi set, hopefully maybe it can be the same, but it probably changes something.

Student: (Refer Time: 11:29).

Right and single key you can have different values, you can probably have even the same pair repeated.

Student: Same key.

Yeah, you can have same key multiple times, same one key multiple values and so on in this output.

Student: (Refer Time: 11:46).

No, it is that is why it is always like that. So, the formal the not the formal, I would say when map reduce was originally designed as a defined as a programming model it was defined like that. So, it is very easy of course, you can cast anything as a key thing right key value, for example, you can you can if you want input a graph a trivial ways just put key as a dummy key and just put everything in the graph. So, you can sort of we can ignore that, but generally we use the key value in a specific, I will come to that in a second. So, meaning that it is does not does not lose generality.

So, you can basically encode anything as the key value pair. So, that is the map phase. What is the shuffle phase? So, the shuffle is nothing, but it moves the shuffle, takes the output of the map phase that is these key value pairs and sort of routes it to machines. So, map of course, map phase is done by machine. So, again, but they are called mappers, mappers are machines they are the same machine, it is not generally people use mappers the refer to the machines as mappers, when they are doing the map phase and of course, is the same machines might be doing the reduce space. So, then they are called reducers. So, during the map phase the mappers compute something they have regardless output of key value pairs and the shuffle space routes these key value pairs to other machines, they are called reducers it can be same machines, but some other machines and the reducers.

So, will basically these are routes. So, it routes these key value pairs. These all upon not one lot that many list it is list basically. A list, it routes these list it will put it as a list to

machines in a certain way, we will come to that we will always there is a does a way to route that come to that second, but it does this routing and what is the reduce phase do. It reduces to the machine. So, so these machine machines are the reducers it is it routes at to the reducers which again they take a list of key value pairs basically this pairs. So, that is why this is completely transparent. So, many times we just completely omit it, when you come to designing algorithms we will see what we what how we explicit state them algorithmically.

So, it takes these key value pairs and then returns again a multi set of some other maybe key double dash value pairs and this again will end one round, and this will be the input for the map phase for the next round. So, these will be routing these again. So, these machines themselves will send it. The reducers will do this computation reduce it. So, this is called the reduction step. So, it will reduce it and this will be automatically routed to the machines for the next to the mappers of the next step and in the process continues. So, that is basically the computational model, is a very simplified computational model. So, why he was defined like this was because I completely have to. So, even shuffle phase I sort of we will see that the shuffle phase is a sort of transparent of the programmer or the system or the algorithm designer.

So, you only has to write the code for the this and this and that is why that is one of the reasons why a dupe and map reduce are being very successful because I just have to write a code for map and code for reducer and then just speed it. This one important thing that is sort of not probably obvious yet, but implicit in the way I defined these operations. These map phase of course, the reduce phase, but especially a map phase there it is completely stateless and that is very important. So, they are stateless. So, all these have stateless, what do we mean stateless? So, every time I am getting a list of key value pairs and just operate on that I do not. So, map machines do not remember what happened previously. So, what is the big advantage of being stateless?

Student: (Refer Time: 16:20)

Yeah. So, you can just schedule the key value pairs independently right. So, they can be as many mappers as possible whenever I have key value pairs and do some mappers. So, they if I do not have to worry about, I have to send this particular key value to that machine because that machine remembered something about it and so on. So, that is why

I get really, if something is really paralyzable this really works. So, that is the big success of map reduces, so that is why it is all about. Google they had this very nice thing of abstracting out otherwise it is not a big thing. So, already it is now the parallel model.

One more thing that is that is that is all typically used. So, one of properties of map reduce is that. So, this mapping is there, but when it when you do the shuffle phases this routing, what happens is it is not these key value pairs are not generally routed to arbitrary machines. It is always routed with the following property. So, all the key value pairs with the same key are routed to the same machine. So, that is the property that is that is other you can put it as part of the model.

So, you have a list of this key value pairs of course, if there is only one unique key then of course, there is only one unique key, but there in the list of key value pairs it is some there can be many pairs with the same key all of them will go to the same machine the shuffle automatic takes care of that, the shuffle is transparent it make sure that all the pass the same key value are routed to the same reducer. Basically, all them let say if there is pairs which are, let us k key k 1 let say this is value 1, k 1 value 5, k 1 value 10 and maybe there are other pairs in the list maybe k 2 value 2, k 2 value 1.

So, you have list of these, let say this 5 pairs which are the output of them map step the shuffle will do, what it will reduce these route these 2 to 1 machine, let say m 1 and route all these k 1 pairs to machine m 2.

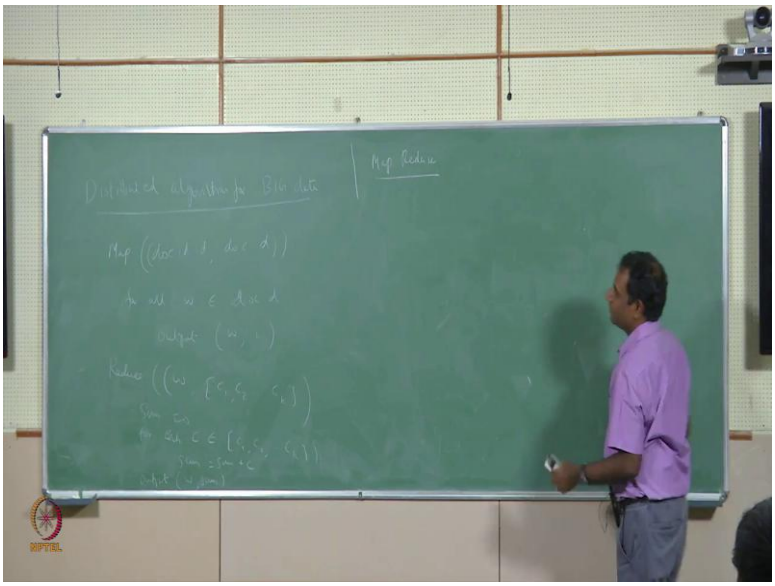
Student: (Refer Time: 18:58)

m 1 can be equal to m 2, that is fine, that is true, but they should go to the same machine, but of course, in point is there. So, large that there will be many machines this piece. So, that is the model, it is a very simple model and let us to understand this probably let us look at a very simple algorithm where this sort of the usually the first algorithm that people sort of give to understand the thing from map reduce and that is what is called let say and this is one of the first applications I think that probably motivated Google right. So, Google let say they want to. So, when you do a search you want to let say you are searching for a word, what Google does it lists all the web pages that word has right that is the one of the basic things it does so obviously, Google designers, the guys who developed it, I think they have won a lot of awards (Refer Time: 19:58). So, they are looking at this and when I think when the web pages they are not too many web pages

probably easy to do it in a few machines, but as the web grows you have to basically process not just the web graph, but every document in the web, you have to process the every web page.

So, you have to read every word in the web page and you have to sort of reverse index it. Let say you want to count the number of words in every document, you have a bunch of documents may be all the web pages in the in the world in the web and you want to count the freq, how many times a word occurs in every document for overall over all the or a bunch of documents you are given a bunch of documents you are you want a given to list all the words in all the documents and the total sum that is the number of times it occurs in all the documents. So, you can write easy map reduce algorithm for this. So, what will the map or reduce algorithm look like?

(Refer Slide Time: 21:20)



So, we can write a map reduce, let say, the map will take as an input that is let say. So, maybe a list of, this is the input this is the actually a list think of that. So, it is a list of all the documents, the document id and the document itself. So, it is a list of all documents. So, I want to count all the words and maybe a bunch of million documents. So, that is a list. So, and it is going to take a list and emit a list and output a list what will be the list. So, it is going to that, all words in this, what is going to do it is just going to output it is going to output a pair. So, it is going to be it is a very simple pair in this case it is going to output this pair.

So, it is going to look at all it is going just. So, this is a very dummy thing right it is going to just read all each it is going to be each document and in each document it is going to read every word for every word is going to output this pair the word and one basically means that that word occurred one time that is it. So, it is a very simple mapper. So, it is maps towards to 1. So, this is a huge list of key value pairs that is the mapper. So, what should the reducer do of course, the shuffle is transparent; that means, what it will reduce it will route all the words which are the same to the to one machine. So, what we will do. Let me look at reduce now. So, I want to reduce the reduce code. So, it will take what it will take a list of pairs. So, maybe I should say like this is a list actually. So, so it is a list of pairs w and what. So, let me write this. So, will be a count ok.

So, it will be list of counts. So, let me write this like this that means what it will take pairs like w comma c 1, w comma c 2 and all. So, of course, this c 1, c 2 might be just 1. So, it takes w 1, w 1, w 1 for many ones and what you will do. Yeah, so maybe initialize sum equals 0 for and then finally, it will output w sum, it will output that pair. So, the reducer simply adds all the. So, I have written generally see, but in some times it can be generally see, but it is usually 1 in this cases is just one why because is the whole and how many rounds does it take one round because there is one map and one just. So, that why it is very efficient. So, that is why for such jobs map reduce is still a way to do I mean if you want to do this kind of. So, map reduce we will see turns out to be not, efficient when you wanted do many rounds because every time you are doing this, what is the sort of the complexity of this kind of a model? What is the complexity of computing in map reducing? What is the sort of the bottleneck?

Student: Shuffle phase.

Yeah, shuffle phase because that is because as I told that principle is very much underlies the principles that we have been seeing all along. So, the map computation and reduce computation are local computation. They are within machine computation they are significantly cheaper. So, the real costly operation in map reduce is really the shuffle phase, because you have to move all these pairs to the reduced machines. In fact, this can take days actually, I think maybe through web page competition will take one round, but just moving this data is. So, thing I mean that is the costly operation just moving them.

So, even the bandwidths are high speed links, but these data are petabytes data. So, they

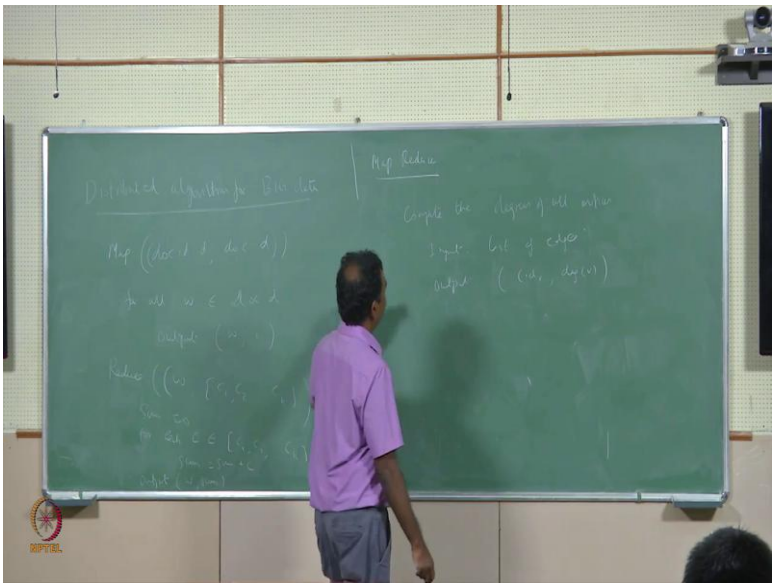
are just moving all the pairs. So, you want to really minimize the number of communication rounds. So, mapping it is really good when you solve problems then like essentially constant number of rounds and it is reasonable and many problems are like that you can solve in constant number of forms. We will see someone say even connectivity there it is not a clear. So, there is a very famous one problem in map reducers is that it is not clear whether you can solve connectivity in less than log n rounds.

So, there is the very easy connectivity algorithm that you can solve in log n rounds, but it is already log n is too much. So, there, map reduce is not suitable then you have to use k-machine model. We will see why that is a better model in their entry, but let us let us go and do one more non trivial thing. So, this is the very like toy thing. So, let us do a real problem. So, let us look at the problem that I have seen a lot the sort of the touch tone. In fact, it is a graph problem. So, maybe before that let us go to even a slightly just to get used to map reduce way of thinking.

So, let us look at, for example, I am given a graph and I want to let say output compute the degree of all the nodes. So, what is the input for this problem again you can think about it is the input you can always think of the list of key value pairs. So, we can think of the input as the list of edges. So, maybe, just an edge and maybe one or something like that because it always should be a pair right, because you can say. So, e 1, because we can if the value can be dummy, for example, so that is why you can encode anything. The value can be dummy or maybe you can just say the I the value will just can be the graph id or something like that. So, if you want to look at one graph that id will be the same. So, that does not give you anything. So, you can take the input as a list of pair edge pairs and output should be a list of pairs again, where the first key will be the node id and it is degree.

So, this is basically like the word the count problem we saw, but it is the count problem in the graph setting.

(Refer Slide Time: 28:34)



So, I want to compute the degrees of all the vertices. So, the input is list of input, the list of pairs, list of edge pairs list of edges as list of edges and the output is again list of pairs where it is basically the. So, id of the vertex idv and its degree v, it is a list for all the vertices that is that is the computation. So, how can I do this using map reduce? I claim that I can do it in basically one round.

Student: (Refer Time: 29:16).

Yeah right.

Student: Is the same.

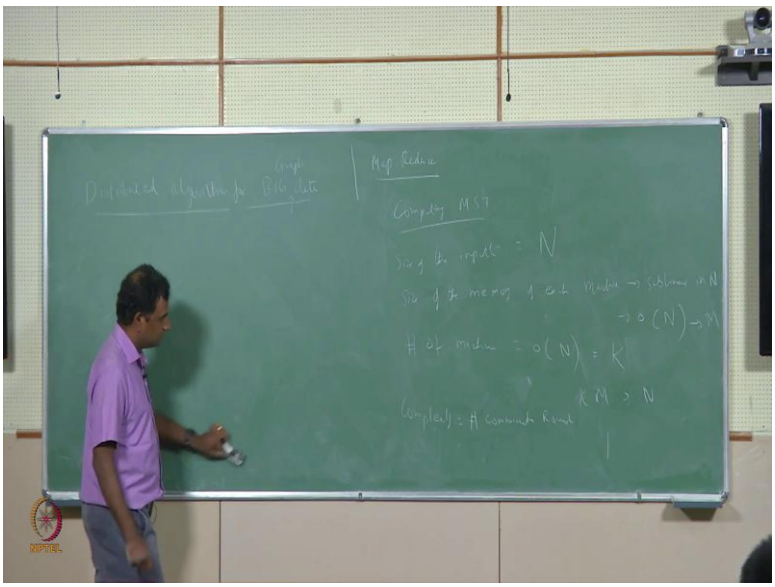
Yeah, same thing basically, this is just using the word counting to do the thing that is why. So, I am just translating back to the word problem, this is just I can do 1 round. So, so why I am saying it like that why I am saying this because I am going to look at non trivial graph format of MST. So, sometimes when we were when we design algorithms for map reduce we do not even go to the level of map and reduce. We sort of, we operate basically we give a code at the machine level, we are a even more little bit more abstract because the lonely with details can I can be implement like this. So, for example, I will say compute the degrees, but I would not actually give a map and reduce thing for that you can do it.

So, when you when you write pseudo code for map reduces it is better to think of not in the level of map and reduce, but at the level of machines. So, machines do something and

they and the shuffle will automatic routes the thing. So, always say machines so each phase can be thought of the machine level. So, you write the code for machine and you say you give the input for the input the pairs for input value for the machine and output value for the machine the output value is automatically routed to the some other set of machines which are the reducers. So, you do not exactly specify how it is routed, you can assume that there is this kind of shuffle function that takes care of it. So, you only talk about in the terms of, you think of it in the writing code in terms of the machine level.

So, of course, you typically only say the pseudo code for one round and of course, you iterate it. So, let us formalize this little bit especially with regard to graph problem because I am going to really look at graph problems. Today, although you can try to big data for anything, I am specifically I am going to look at today big graph data although it is not because it some keeping with you will see that is why I am the connection would distributed algorithms the kind of stuff that that we are being seen till now.

(Refer Slide Time: 31:38)



There is a huge connection that is a very close connection. So, let us look at, right now I am going to look at our usual problem, I want to compute, I want to write a map reduce algorithm for computing MST that is my goal, but before that I want to really give a mod. So, what the model I gave you this map introduces really a specification model. So, that is the model that the programmer kind of users. So, map and reduce, but I have not really told out how to compute the complexity of it.

Of course, the complexity is the number of rounds, but I want I did not say anything about if I do not say any further actually you can trivialize everything I told you, of course, this will take one round, but if I do not say anything further anything further about the execution model you can say trivially I can solve any problem in one round why because you take the huge problem and then just you can just you can just you can encode a huge problem with just one key pair and just route it one machine; that means, route all the data put one just one key for all the data and that is will routed will be routed to one machine and that will locally solve it and that is it. So, everything can be solved in one round. So, that completely trivializes map reduce that does not capture what is happening in practice that is not happening like that.

So, what I mean, for example, in this work I can put everything as one all documents of one key value and then all the documents come to one machine and that machine locally computes the things, but that is like as central computation. So, of course, my complexity model does not disallow that. So, you need a complexity model that also captures what is happening in practice that basically that non-trivializes the thing this trivializes thing. So, this is sort of the mapping, I told you sort of the specification model. So, I need a complexity or an execution model that sort of captures, what is happening that captures the big data.

So, I am not till told you what is the big data aspect of the model. So, why is it really big data and that is this. So, basically I am going to assume that let say that is all the size of the input because I want to do complexity, obviously, it has to be depend on the size of the input here it is basically the list of the input key value pairs.

For example, it is a graph it is a number of edges in the graph. So, it is a huge graph that is a lot. So, basically m is the size number of edges. So, consider this is, assume to very huge. So, I do not want to trivialize the problem by routing all the edges or the entire graph in one machine then it is trivial. So, what is the constraint I should put to make this whole model non-trivial? So, I told you the main thing about big data as I do not have, I cannot fit the old data in one machine. So, I have to put that constraint.

Yeah, even because that it is not, say n. So, even that is number of edges. So, this is n negation of quality n, do not think of n as the number of nodes. So, this is the size of input in the graph it is the number of edges, other things maybe I want to sort a bunch of

things or whatever then it is a number of numbers or whatever. So, it can be, let me do not think of that is I am writing as big n let me the size of the input the graph setting it might be number of edges. So, the main thing, assume is a constraint on the memory of each machine that is the main thing that makes the whole thing meaningful.

So, that is the constraint on the. So, this is a bunch of machines I come to that, but size of the memory is a most important thing. So, there is a bunch of machine right these are both these that is why I only talk only at the level of the machines it can be a map or reducer or whatever, but whatever be the machine. There is some map reducer there is a constraint of size of memory of each machine. So, we will always assume that this is sub linear that is the that is the thing that people are assuming I mean it is significantly smaller than this input size; that means, I cannot fit the whole data in one machine that is just impossible.

So, that rules out immediately rules out this kind of trivial things like one round it is not possible because I cannot even fit every things, I cannot route everything in one machine so; that means, it is typically sub linear. So, little typically little o of n then always not typically always little o of n and also the number of machines is also a sub linear. Of course, there should be one thing right the number of machines times the memory of one machines should be at least t equal to the total size because I have to fit the total thing somewhere of course, with that constraint. So, still if you look at the number of machines let say the number of machines is k and this is the memory is m, of course, k times m should be otherwise I cannot fit everything in that thing right. So, I should be able to fit.

Student: Lots of times we should have some (Refer Time: 36:53)

Yeah, otherwise yeah that is true that is. So, initially it is distributed somewhere, but map reduce algorithms, if we look at it let say I have an algorithm that terminates in constant number of rounds; that means, what the algorithm must finish this job without any machine seeing the whole input because there is no even time to see the whole input because each does it see only little often it is finish interons. So, no machine as saw the whole input, but still the computation as finished that is the situation I am that is why I constant rounds make sense or definitely were little rounds. So, that is the idea, you. So, you basically finish your job without any machine seeing the whole input. So, will come to an example, but that is the computational model.

Now, what is the thing I have to reduce, what is the complexity? Now, again it is the given this constraint I want to reduce the number of rounds; that means, each round is this three things which is transparent because now I am talking about the machine level. So, reduce the complexity is the number of communication rounds you want to reduce this ideally you want to be constant. So, again like local computation we ignore it is polynomial; that means, each in each round the computation done by each machine is polynomial, but we sort of ignore it.

Student: (Refer Time: 38:30)

Yeah.

Student: Because (Refer Time: 38:34) we are just taking memory in our computer what about the time because.

Correct

Student: Mappings can generate a lot of.

Correct.

That can lead to a large amount of.

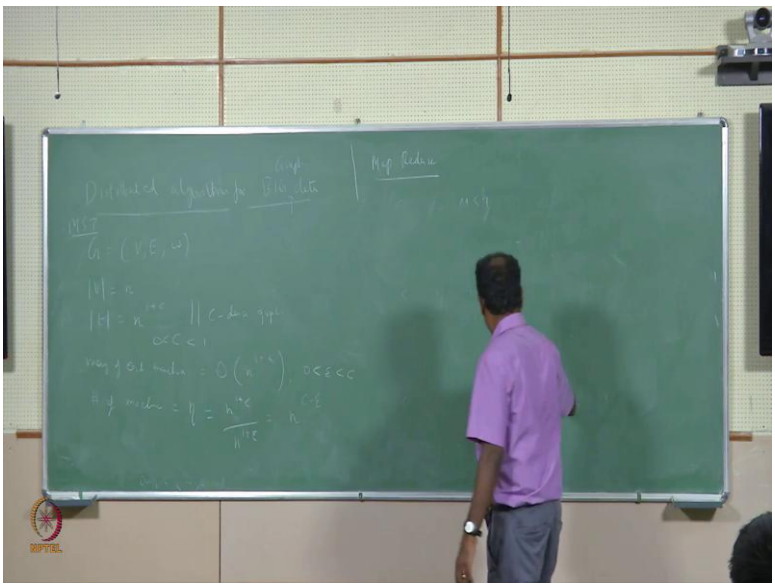
Correct, but that is going to; we are not going a sort of counter that we are not going to charge it to the computation, but we are going to charge it to memory, but then there this larger then it is a problem right may be it is taking too much thing. So, in that sense indirectly, we are reduce we are restricting computation because too much memory is the problem right, but generally it is going to be polynomial we are not going generally we do not assume like the distributed thing I also said the local computation is free and typically we would not abuse it; that means, we do not do non super polynomial.

In generally even do polynomial. In fact, all the algorithms you have seen now they are very simple, it is especially in the congest model they are very simple so; that means, if you think about it. So, many of the algorithms that have seen in the congest model if there were of especially of very small very local computation you can translate them into P RAM model. So, you can think of you can translate them into P RAM algorithms, but not the other way around the other way around is more complicated, but this is

reasonably easy. Of course, if the local computations are very costly then you cannot do that because in P RAM local computation count.

In one round of P RAM, I can only do like small number of very one step of the computation. Now, we are going to look at computing the MST, I want to do come to the map to the map reduce algorithm for computing the MST, again the graph was given us a list of edges and I want to output, again it will be a list of edges, but it will be the MST edges, the MST edges will be output. So, what will be an algorithm? So, again I am going to look up a little bit more want to write explicit because it is, for example, some of these things might involve completely degrees and also those are sub routines. So, those are sort of I am not going to talk about this. So, I am going to assume it. So, I am going to give you an algorithm that works that finishes in constant number of rounds, but it does not come without assumptions. So, I am going to give some assumptions on the graph itself.

(Refer Slide Time: 40:56)



So, let say, I am giving you an input graph; weighted graph. So, let me have some assumptions. So, the number of nodes is n. So, number of edges, we will assume it is called see dense. This is, such a graph is called a. So, I am going to the input see dense graph; that means, the number of edges is slightly super linear of course c can be 1 close to 1. So, for starts you have to do a slightly different kind of anything, but right now let us assume it is so that means, c is. So, why this is the reasonable thing is one of the

reasons people probably study this is many real world large graph like the web graph and all it is sort of see dense the see is like I think it is some point 1 or point 2 or something.

so they are slightly see dense graphs the large graphs that you see in real world they are not completely is sparsed, they are slightly super dense slightly dense. So, that is the thing. Now, this is the input right of course, weights are there on the edges, there are it is of. So, I am going to do MST and of course, by the memory constraint I am going to assume that the memory of each machine. So, it is what is it has to be sub linear. So, I am going to assume it is to n to the 1 plus epsilon, but of course, epsilon should be strictly smaller than c. So, it is still sub linear and I am going to assume.

So, it is important why I make this assumption. So, it is definitely sub linear, but note that I am assuming not end to the epsilon or I mean I am not assuming end to the 0.9, I am always assuming n to the 1 plus. It is very important do that without that it is there is you cannot get the kind of algorithms that I am getting constant why what is the motivation for doing allowing this. So, it is still sub liner that means, what no machine can hold the entire it non-trivializes the problem.

Student: In two rounds.

I am not, I do not know about that, but may be not 2 rounds; we show we can give constant round, but that is not the main that is. So, of course, without that assumption I cannot let say little of n here that is also sub linear right, but if I say that; that means, what no machine can even look at all can hold all the nodes basically that becomes very difficult. So, it still where it has still open to get efficient algorithms in that model. So, it becomes more difficult to that.

So, it is not the easy algorithm that I am going to tell you, but still it is a very nice algorithm it is very simple, that means, this allows each machine to hold all the nodes because there are at least n memory space I can at least hold all the nodes, but not all the edges of course, they hold all the edges I can send it to that machine and it is wonder. So, I cannot hold all the edges because this is strictly smaller than c. So, think of c as like 0.2 or something an epsilon is may be 0.01 it is slightly it might be slightly larger than 1 slightly larger than 1 may be 1.01 that is the memory. So, what is the number of machines, of course, the number of each memory times of the thing should be at least the thing. So, let say the number of machines is eta. So, eta should be, may be let say we can

assume it as they can assume that it has, let say n to the 1 plus epsilon by c.

So, this is n to the there are too many more than, many you do not need more than. So, many machines if you have more than you ignore it because I am not going to use it right because intuitively I can I need to use only as many machines that has the memory right. So, since I have each machine as so many memory, I will always try to use it fully because that whole point is memory right. So, I will try to use if machine has some memory, yes I am going to all the memory, I am going to fill the bit sort of after that I am going to ignore I have addition machines I do not care because if you have the memory use the memory. So, that is the model.

So, in this model I want to give a constant because generally that is where it is good you can already see that the map reducer is good, but you can see there are lot of assumptions, here if already for this does not what about sparse graphs and things like that.

Student: We need to the epsilon to be greater than 0 great this files that hold all n vertices and may be NHs.

No, it I really need to be greater than it cannot be 0, it cannot be 0 let see that think I think if it is 0 means that number of rounds will be infinite I think. But let me given if what is the good map reduce algorithm in this I want to come up with the actually ideally I told you map reduce algorithms are good only that constant round algorithms even super constant is not very good at least in definite in practice of course, theory it is good, but definitely will not in practice, and even for that is why we have to assume. So, that is one reason. So, this is people I am assuming super dense (Refer Time: 47:09) and super liner memory, super linear in terms of n term.

So, let us look at an algorithm. So, let us think of a (Refer Time: 47:19) algorithm for finding mapping MST now, I am going to do MST. So, how will I solve MST? So, note that since I am shooting for like constant rounds no machines I have seen the entire graph, but still there will be able to compute the MST. So, what I can do. So, I am going to exploit crucial properties of MST it is actually very simple algorithm. So, in fact, it is in some sense the obvious algorithm what will I do. So, of course, what is the input? The input is the list of edges and graph is given as a list of edges.

(Refer Slide Time: 48:04)

What should I do? So, I have some machines initially let say at any point I will have enough machines to hold all the edges initially it will be eta. So, I am what I am going to, so the algorithms idea is very simple. So, keep on provening the number of edges in the graph till I have the number of the list of edges is small enough to fit it into one machine and then solve the problem locally in one machine that is it

Student: what is the expected output out of these?

The MST

Student: In what form?

It will be a list of, we again it is a list of in the map list setting it will be a you can say a MST, but it will be a list of edges, MST edges; n minus 1 edges.

Student: All in this one machine.

Yeah, it does not matter it just emitted. So, I will it is may be printed somewhere (Refer Time: 49:03), different machines can emit different things, but here in this setting it can be the same machine because that is where I have assumed that; that is why it is so crucial that I have assumed that one machine can hold super linear memory, super linear in number of nodes. So, one machine will presumably, so I think the algorithm that I am going to describe the one machine final machine which holds the thing will output the

entire thing. So, it will print the all the edges. So, it will have that, it will all the MST edges it can hold that.

So (Refer Time: 49:42) more complicated if you say that each memory each machine has little of a memory then different machines have to output different edges. In the k machine model you will see that; that is what is going to happen. That is more realistic and also for different cases and we will come to that in a second. But the idea is very simple. So, the main technique I will say here is filtering, main algorithmic technique. So, it is very natural thing right I keep on filtering the number of edges, filtering just provening the number of edges, so that whenever I saw a list of edges in the second round I will have some other list of edges; the third round some other list of edges. Of course, all these lists will be reduced by a large factor.

With the property that MST edges are still contained here. So, the edges that I am provening away are guaranteed to be not in the MST; that is the only thing I need to do. So, what will I do? So, the algorithms is very simple let me just describe it. So, I will always have some list of edges at some round. Initial it is the entire list of edges; initially I will have all the machines in my disposal because I have so many machines, eta machines because that is enough to fit all the things. I am going too randomly, so the main algorithms it is a randomized algorithm. So, the main thing is the randomized algorithm. So, it is going randomly send each edge to; it is picks a edge and since it is your random machine. So, every edge is assigned to a random machine.

Now once I get one (Refer Time: 51:23) machine gets its machines what will it do? It will compute the minimum spanning tree or forest of course, if it is computes minimum spanning tree then it is done. If it computes n minus 1 edge it is done, but in generally it might be a forest. So, initially I will have n to the 1 plus epsilon, so they are sent randomly to different machines, eta machines. So, each edge is send to one of the eta machines randomly. So, let say this is m 1, m 2, m eta. So, each will get exact, so this is fine right because I know that since I have set up eta like that each will get at most n to the 1 plus epsilon edges that is fine, it too fix in the memory. So, what will it do?

Student: On that is sub graph.

On that sub graph it will locally compute MSF a Minimum Spanning Forest, how will it do?

Student: Doing (Refer Time: 52:35).

Correct, it will just use the cycle property, just using the cyclic property it will just remove all the; those edges are guaranteed to be not in the MST. That means, you look at all the cycles remove the heaviest edge in the cycle, in all the cycles in the graph that is just a local thing. Then how many edges it will output?

Student: n minus 1.

Correct, it cannot be more than.

Student: less than n minus 1.

Yeah, it cannot be more than n minus 1. So, each one will output n minus 1. So, how much I have reduced, what is the factor I have reduced the number of edge. So, initially each machine had n to the 1 plus epsilon and each machine have reduced by a factor of n to the epsilon n to the epsilon. Now, what should I do the next thing?

Student: Reduce the number of machines.

Reduced the number of machines, so I always take the number of machines to be the total number of edges divided by the memory, so the number of machine l; that is initial it was like this. Say number of machines, number of edges currently there, right say number of edges in round i divided by the memory because the number of each machine is, the total number of edges.

So, these guarantee what? Every time in the number of edges in each machine will be what? N to the 1 to the epsilon, because I am dividing by that because that is the memory of each machine; so sorry number of machines sorry I should say yeah. So, that is right this is the number of machines, but this is the memory of each machine. So, this is the number of machines. So, this is the memory of one machine. So, every machine will get, in every round will get how many edges? n to the 1 plus epsilon and it will reduce it to.

Student: n minus 1.

So, n minus 1, so in every round what is the factor reduction?

Student: n to the power of

N to the epsilon and I am starting with n to the.

Student: (Refer Time: 55:17).

So, every time I am reducing by n to the epsilon how many rounds it will take?

Student: Constant number of rounds.

Constant number of rounds, because every time I am reducing epsilon

Student: (Refer Time: 55:26).

Minus epsilon, minus epsilon; c over epsilon.

Student: C over epsilon

C over epsilon (Refer Time: 55:30) 1 plus c over epsilon some (Refer Time: 55:33) it is a constant number of rounds because c and epsilon are constants that is the algorithm; very simple, this algorithm is used in Google. So, the guys who came up with this algorithm are Google people. But I will give you a better algorithm in the k machine model. But before that, but what is the only thing that we have to analyze here. So, this is guaranteed, when this is guaranteed? So, I need one thing to guarantee this; the randomization step because I just have to show that the randomization step is balanced. So, because the point is the red there; so, the mapper does the very simple thing I just said it abstractly, what does the mapper do? Just takes the edges and randomly sends it to different machines.

So, the mapping is not balanced then I do not get too much because I want the mapping to be balanced. That means, everybody gets n to the 1 plus epsilon, so it is a good reduction. So, I just have to make sure that there is a good reduction.

Student: It an arbitrary distribution of edges machines because any ways spanning (Refer Time: 56:47) bridges will be at most (Refer Time: 56:49).

That is true, that is true, but the point is I want to do this quickly though. So, I really want to do this quickly in way that confirms to this. So, maybe I did not say one thing, one thing that may be this is something that you are right. So, one thing that I forgot to say and why this randomization comes naturally is because this; let us go back to the

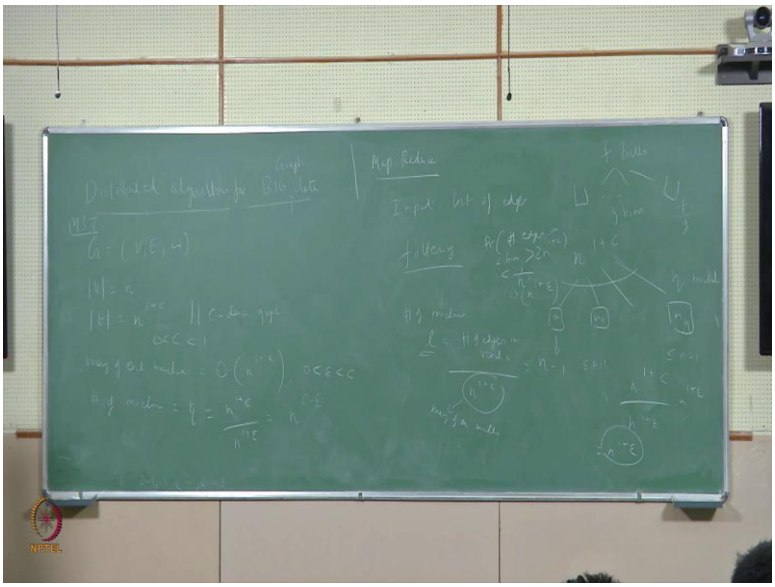
specification model that we saw before. Like there is this map shuffle and reduce. So, the shuffle takes what it takes the key value pairs with the same key and routes it to the thing. So, there can be multiple that can be lot of key value pairs with multiple same keys. So, let say key one bunch of pairs key two bunch of pairs.

So, what it does is; in general I do not know what the load is and all. So, what it does is it basically hashes it. So, each key each key is hash to some machine; that means, that means I use the hashing function. So, we are going to assume that this hashing function is a random because why is that because I want keys that are generate it same key that might be generated by a different mappers to go to the same thing. So, there is no time to talk and coordinate and all; there is no time to talk and co ordinate. So, I did not say that, but generally the thing this is done by a hashing.

So, there is a hash function a share the randomness, the hash function gives you shared randomness; that means what? It is random enough. So, it is usually the best way to do it by I mean of course, in practice you use the hash function, but it is theoretically the best way to do this universal hashing use a universal hash function. That means what? It is hashed that means, it is if I take one key it is there is a unique output and it is done without any coordination. So, the machines cannot coordinate and say this thing I mean let us agree that all the key values, that all the values of the same key should go to this particular machine, there is no time do that, there is no such facility in the model to do that. So, it has been done instantly. So, the hashing is the beautiful way of doing it

So, I just take the hash it to the key and get the messed. So, hash will give you the output of the machine id. So, I (Refer Time: 59:16) to that machine. So, everybody is (Refer Time: 59:18) it. So, that is why I am using a random value. So, the randomness also comes naturally. So, everything is here hashed randomly every edge. So, here every edge like that is the key; that is one key for that separate key, each edge is the separate key. So, you just have to show that; so this just a load balancing problem. So, if we have let say f items and so there are whether you know balls and bins. You have seen balls and bins.

(Refer Slide Time: 59:55)



So, you have f balls and let say you have g bins, so there are g bins. Think of there are f balls you have to throw the balls randomly the bins, assume that each ball is thrown randomly in the bins that exactly what is happening. The balls of the edges, the bins are the machines. So, one way I have to do that is; that is why I mean that is why I am not doing or it is not very efficient to do what you are saying for example, of course, if ideally I want load balancing, so that means I want approximately every bin to get f by g balls. So, one way is do it, is to do it sequentially. So, put the first f here second f here; that means coordination.

Student: (Refer Time: 61:46)

Correct, that is why I do not do it; that is why I do hashing. It is very powerful, random variation. So, there is no coordination, but it gets you on average what is the average if I do; that means every ball is thrown to random bin. So, take a ball throw to a random bin; that means every ball has probability of 1 by g to go a particular bin. So, you did not see what is there in the bin, just throw it. So, this completely state less, it is no coordination, nothing. What is the expectation? How many balls?

Student: F by g.

F by g and you can also show the i probability it will be f by g. For example, here in the very first step there are so many balls n to the 1 plus c balls; number of bins is eta there

are eta machines. So, eta is what? eta is that so that means, n to the 1 plus c divided by this eta which is n to the 1 plus c times n to the plus times n to the 1 plus epsilon which is basically n to the 1 plus epsilon.

So, on the average I am perfect that means, I am going to get 1 n to the 1 plus epsilon, but if the deviation is too much then it is bad right let say one machine gets n to 1 plus epsilon dash, where epsilon dash is slightly larger; that is not good, that means I cannot store so many edges. So, that is a bad thing. But what I can show is that this is the expectation, because this is the f by g is the expectation that is well congregated. So, I am not going to, but it is very easy to show by Chernoff model; that means what? With i probability the number of number of edges in bin being larger than let say, so the expectation is exactly 1 to the n to the 1 plus epsilon; being greater than let say twice that is very small let say it is very small. Can easily show this by just using probability (Refer Time: 62:56) bounds.

Student: It divides the properties of the hashing (Refer Time: 62:58).

Correct, right now I am using I am being little bit more just; I am going to assume that it is random hash function; that means, it is completely random, but that is not quite true though. But you can simulate that, you can assume, for analysis we will assume that its basically throwing balls randomly, but that is why, but there is a slight technicality that is when you analyze the for simplicity it will assume this. But if you really think about it hash function is not exactly random because it is not really random, it is deterministic function right, but you can show that you can design hash functions such as universal hash function that are essentially like random. So, they have this big point.

But for analysis we just assume that they are random; that means, they are independent, but they are not independent right it is a function so every key has go to the same thing it is like saying that balls of certain type will always go to same bin, but that is not true here in balls case, but in hashing case that is true. So, it is not exactly random. So, deterministic, that the hash function is deterministic it is the one function, but it has this property that the items chosen are; that is why is called universal hashing, you have to use universal hashing; there is enough randomness there to get this property. But the (Refer Time: 64:10) in adjusting them as like independent.

Student: So, edge is never assigned to (Refer Time: 64:16).

Yeah, only one machine; the only problem is that is clear because they always send it into one machine by hashing, but I do not want. So, here also here one ball is always send to one bin, but you do not want to some bin to be overloaded. For example, some machine to get number of edges which is much larger than the memory because the memory is fix to be like I would not say n to the epsilon, but I mean I should say instead o of n to the epsilon right. So, what I am showing is since if I fix the memory to be 2 n (Refer Time: 64:45) 1 to the epsilon with i probability there will be no violation; there will be no violation.

So, that slack is given by the laws in randomness, you can show that slack it is not big thing. So, this is the algorithm for map, for who want to do MST in map reduce. So, this is the algorithm you will write in a dupe. So, you are (Refer Time: 65:15) actually I am giving it is an abstract way, but you have to write the map and reduce that explicitly, that is not difficult.